KERJA PRAKTIK - KI141330

Pembuatan Aplikasi Web E-commerce SURAJ

SULTAN AR RAJABI (SURAJ)
Barat Sidogiri KM-2 Kanigoro-Rembang, Pasuruhan

Periode: 25 Agustus 2020 - 30 November 2020

Oleh:

Jeremy Vijay Wongso	05111740000062
Pristi Zahara	05111740000112
Rana Wijdan Naim	05111740000187

Pembimbing Jurusan
Ary Mazharuddin Shiddiqi, S.Kom., M.Comp.Sc., Ph.D
Pembimbing Lapangan
Dr. Radityo Anggoro , S.Kom, M.Sc.

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

[Halaman ini sengaja dikosongkan]

KERJA PRAKTIK – KI141330

Pembuatan Aplikasi Web E-commerce SURAJ

SULTAN AR RAJABI (SURAJ)
Barat Sidogiri KM-2 Kanigoro-Rembang, Pasuruhan

Periode: 25 Agustus 2020 - 30 November 2020

Oleh:

Jeremy Vijay Wongso	05111740000062
Pristi Zahara	05111740000112
Rana Wijdan Naim	05111740000187

Pembimbing Jurusan
Ary Mazharuddin Shiddiqi, S.Kom., M.Comp.Sc., Ph.D
Pembimbing Lapangan
Dr. Radityo Anggoro , S.Kom, M.Sc.

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

KERJA PRAKTIK

Pembuatan Aplikasi Web E-commerce SURAJ

Oleh:

Jeremy Vijay Wongso	05111740000062
Pristi Zahara	05111740000112
Rana Wijdan Naim	05111740000187

Mengetahui,	Menyetujui,
Sultan Ar Rajabi	Dosen Pembimbing
Pembimbing Kerja Praktik	Kerja Praktik
Dr. Radityo Anggoro , S.Kom, M.Sc.	Ary Mazharuddin Shiddiqi
	NIP. 198106202005011003

[Halaman ini sengaja dikosongkan]

Pembuatan Aplikasi Web E-commerce SURAJ

Nama Mahasiswa : Jeremy Vijay Wongso
NRP : 05111740000062
Nama Mahasiswa : Pristi Zahara
NRP : 05111740000112
Nama Mahasiswa : Rana Wijdan Naim
NRP : 05111740000187
Departemen : Informatika FTEIC-ITS
Pembimbing Jurusan : Ary Mazharuddin Shiddiqi
Pembimbing Lapangan : Radityo Anggoro

ABSTRAK

Suraj (Sultan Ar-Rajabi) adalah sebuah usaha baru yang berdiri pada tahun 2019, didirikan oleh tiga orang, yaitu Luchman Ali Syahbana, Isa Nurudin Ali, dan Musa Ali Perdana, yang masing-masing sudah malang melintang di dunia usaha. Suraj bergerak di bidang bisnis syar'i, berbasis pada sektor pertanian, peternakan, perikanan, perkebunan, dan Franchise Kuliner. Dengan banyaknya bidang usaha yang digeluti oleh Suraj dan target pasar yang luas, mereka membutuhkan suatu aplikasi web yang dapat diakses dengan mudah oleh seluruh target pasar mereka, sehingga dapat menjual produk yang telah diproduksi dengan mudah atau biasa disebut dengan toko online.

Aplikasi ini dibuat dengan menggunakan bahasa pemrograman web seperti PHP, HTML, CSS, dan Javascript dengan menggunakan DBMS MySQL Server. Aplikasi ini nantinya pengguna

diharapkan dapat melihat katalog produk yang ada di Sultan Ar-Rajabi serta melakukan pembelian pada produk-produk yang dikehendaki, sedangkan pemilik perusahaan dapat mengelola produk serta pemesanan dan transaksi yang dilakukan oleh pengguna.

Kata kunci: Jual Beli, Aplikasi Web, Toko Online

KATA PENGANTAR

Puji syukur kami haturkan kepada Allah SWT karena berkat rahmat-Nya kami dapat melaksanakan salah satu kewajiban kami sebagai mahasiswa Departemen Informatika, yakni Kerja Praktek (KP).

Kami menyadari masih ada kekurangan baik dalam pelaksanaan kerja praktik maupun penyusunan buku laporan ini. Namun, kami berharap buku laporan ini dapat menambah wawasan pembaca dan dapat menjadi sumber referensi. Kami mengharapkan kritik dan saran yang membangun untuk kesempurnaan buku laporan kerja praktik ini.

Melalui buku ini, kami juga ingin menyampaikan rasa terima kasih kepada orang-orang yang telah membantu, baik langsung maupun tidak langsung, dalam pelaksanaan kerja praktik hingga penyusunan laporan. Orang-orang tersebut antara lain adalah:

1. Kedua orang tua penulis.
2. Bapak Ary Mazharuddin Shiddiqi S.Kom., M.Comp., Ph.D selaku dosen pembimbing kerja praktik.
3. Bapak Radityo Anggoro, S.Kom., M.Sc., Dr.Eng., selaku koordinator Kerja Praktik sekaligus pembimbing lapangan kami di Sultan Ar Rajabi.

Surabaya, November 2020

Penulis

[Halaman ini sengaja dikosongkan]

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

1.1. Latar Belakang

Saat ini teknologi informasi berkembang begitu pesat terlebih lagi dengan adanya internet. Membuat masyarakat memanfaatkannya untuk mendapatkan informasi, sekedar mencari hiburan, hingga melakukan transaksi jual beli secara online.

Adanya internet menyebabkan dunia perdagangan mengalami perkembangan yang signifikan. Hal ini menyebabkan transaksi jual beli online sudah menjadi hal yang lazim dilakukan. Saat ini sudah banyak kalangan pengusaha yang memanfaatkan teknologi sebagai senjata utama mereka dalam melakukan proses jual beli, yaitu dengan membuat website e-commerce (toko online). Dapat dikatakan bahwa adanya website e-commerce dapat memperluas target pasar dan melayani pembeli dalam skala besar.

Suraj (Sutan Ar-Rajabi) adalah salah satu pelaku usaha yang bergerak disektor pertanian, peterenakan, perikanan, perkebunan, dan Franchise Kuliner yang membutuhkan jangkauan pasar yang luas dan harus melayani pembeli dalam skala besar.

Dengan melihat adanya kebutuhan tersebut, pembuatan website e-commerce menjadi solusi untuk lebih luas menjangkau target pasar serta mempermudah proses jual beli, penyediaan informasi produk dan pelayanan konsumen dalam skala besar.

1.2.Tujuan

Tujuan Kerja Praktik ini adalah mengimplementasikan fitur-fitur berdasakan kebutuhan perusahaan:

- Membuat fitur yang memungkinkan pengguna untuk melihat katalog produk yang ada di Sultan Ar-Rajabi serta melakukan pembelian pada produk-produk yang dikehendaki.
- Membuat fitur yang memungkinkan pemilik perusahaan untuk mengelola produk serta pemesanan dan transaksi yang dilakukan oleh pengguna.

1.3. Manfaat

Berikut manfaat yang diperoleh melalui Kerja Praktik dalam pembuatan fitur-fitur sesuai kebutuhan:

- Dapat memudahkan pengguna untuk melihat katalog produk yang ada di Sultan Ar-Rajabi serta melakukan pembelian pada produk-produk yang dikehendaki.
- Dapat memudahkan pemilik perusahaan untuk mengelola produk serta pemesanan dan transaksi yang dilakukan oleh pengguna.

1.4. Rumusan Permasalahan

Berikut rumusan masalah dalam pelaksanaan Kerja Praktik pembuatan fitur-fitur sesuai kebutuhan:

- Bagaimana membangun fitur-fitur pada website E-commerce SURAJ?

1.5. Lokasi dan Waktu Kerja Praktik

Kerja Praktik ini dilaksanakan pada waktu dan tempat sebagai berikut:

Lokasi: Teknik Informatika ITS

Alamat: Jl. Teknik Kimia-Gedung Teknik Informatika Kampus ITS Surabaya Jalan Raya ITS, Sukolilo, Surabaya 60111

Waktu: 25 Agustus 2020 – 30 November 2020

1.6. Metodologi Kerja Praktik

1. **Perumusan Masalah**

   Untuk mengetahui domain dan fungsionalitas, dijelaskan secara rinci bagaimana sistem yang harus dibuat. Penjelasan oleh pembimbing lapangan kerja praktik kali ini menghasilkan beberapa catatan mengenai gambaran secara garis besar tentang kebutuhan atau fitur apa saja yang harus ada di dalam website. Setelah mendapatkan gambaran sistem, diskusi lebih lanjut dilakukan guna menentukan rancangan serta *tools* pendukung pembuatan sistem.

2. **Studi Literatur**

   Pada tahap ini, setelah ditentukannya rancangan *database*, bahasa pemrograman sampai dengan teknologi beserta *tools* tambahan yang digunakan, dilakukan studi literatur lanjut mengenai bagaimana penggunaannya dalam membangun sistem sesuai yang diharapkan.

   Aplikasi yang akan dibuat merupakan system

yang akan dibangun, ada beberapa tools yang digunakan. Untuk frontend digunakan Bahasa pemrograman HTML, CSS dan JavaScript. Sedangkan untuk backend digunakan Bahasa pemrograman PHM. Ada beberapa tools tambahan yang mendukung pembuatan website tersebut yaitu XAMPP dan Visual Studio Code. Framework yang digunakan yaitu Laravel dan ReactJS. Database yang digunakan yaitu MySQL.

3. **Analisis dan Perancangan Sistem**

   Langkah ini meliputi penjelasan awal tentang sistem. Bagaimana cara kerja sistem dengan skenario tertentu. Dari penjelasan awal telah didapatkan beberapa kebutuhan fungsional secara garis besar. Kemudian dilanjutkan dengan memperjelas dan menspesifikkan kebutuhan- kebutuhan tersebut. Dilanjutkan berdiskusi dengan pembimbing lapangan untuk mengetahui apakah kebutuhan-kebutuhan tersebut sudah tepat.

4. **Implementasi Sistem**

   Implementasi sistem didasarkan oleh perancangan dan analisis sebelumnya. Penentuan atribut atau fitur yang akan digunakan pada model juga didasari pada analisis sebelumnya. Penentuan tipe data dan format keluaran juga disesuaikan dengan kebutuhan.

5. **Pengujian dan Evaluasi**

Pengujian dilakukan oleh pembimbing lapangan dan anggota tim lain setiap fitur yang sudah selesai untuk memberikan evaluasi ketika ada yang tidak sesuai, dan persetujuan apabila sudah sesuai.

1.7. Sistematika Laporan

Laporan kerja praktik ini terdiri dari 7 bab dengan rincian sebagai berikut :

1. Bab I: Pendahuluan

Bab ini berisi tentang latar belakan masalah, tujuan, manfaat, rumusan masalah, lokasi dan waktu kerja praktik, metodologi, dan sistematika laporan.

2. Bab II: Profil Instansi

Bab ini berisi sekilas tentang profil Sultan Ar-Rajabi.

3. Bab III: Tinjauan Pustaka

Dalam bab ini dibahas mengenai konsep-konsep pembuatan model, dasar teori, teknologi yang dipakai dalam pembuatan model.

4. Bab IV: Analisis dan Perancangan Sistem

Dalam bab ini dibahas tentang proses analisa kebutuhan berdasarkan kondisi yang sesungguhnya dan perancangannya yang meliputi desain aplikasi yang akan dikembangkan. Proses analisa dan desain aplikasi menghasilkan daftar fitur yang dibutuhkan.

5. Bab V: Desain Model dan Implementasi Sistem

Dalam bab ini dibahas tentang desain model dan implementasi secara keseluruhan.

6. Bab VI: Pengujian dan Evaluasi

Dalam bab ini dibahas tentang skenario pengujian, dan evaluasi pengujian setelah model selesai dibangun.

7. Bab VII: Kesimpulan dan Saran

Bab ini berisi tentang kesimpulan dan saran yang didapatkan dari tugas selama kerja praktik.

[Halaman ini sengaja dikosongkan]

BAB II
PROFIL INSTANSI

2.1. Profil Instansi

Suraj (Sultan Ar-Rajabi) adalah sebuah usaha baru yang berdiri pada tahun 2019, didirikan oleh tiga orang, yaitu Luchman Ali Syahbana, Isa Nurudin Ali, dan Musa Ali Perdana, yang masing-masing sudah malang melintang di dunia usaha. Suraj bergerak di bidang bisnis syar'i, berbasis pada sektor pertanian, peternakan, perikanan, perkebunan, dan Franchise Kuliner.

[Halaman ini sengaja dikosongkan]

BAB III
TINJAUAN PUSTAKA

3.1. *E-commerce*

E-commerce (*electronic commerce*) atau perdagangan elektronik adalah penyebaran, pembelian, penjualan, pemasaran barang dan jasa melalui sistem elektronik seperti internet, televisi, dan jaringan komputer lainnya. *E-commerce* dapat melibatkan transfer dana elektronik, pertukaran data elektronik, sistem manajemen inventori otomatis, dan sistem pengumpulan data otomatis.

3.2. HTML

Hyper Text Markup Language (HTML) adalah sebuah bahasa markah yang digunakan untuk membuat sebuah halaman web, menampilkan berbagai informasi di dalam sebuah penjelajah web Internet dan pemformatan hiperteks sederhana yang ditulis dalam berkas format ASCII agar dapat menghasilkan tampilan wujud yang terintegerasi.

3.3. CSS

Cascading Style Sheets (CSS) merupakan mekanisme sederhana untuk menambahkan *style* (seperti warna tulisan, *font*, jarak tulisan) ke dalam dokumen web. Hampir semua *browser* dan banyak aplikasi yang ada saat ini telah menunjang penggunaan CSS.

3.4. JavaScript

JavaScript adalah sebuah script yang memungkinkan kita mengimplementasikan hal-hal kompleks pada halaman web terutama yang bersifat interaktif. Javascript juga dapat digunakan untuk memanipulasi dan mengirim data pada browser pengguna.

3.5. PHP

PHP pada dasarnya merupakan singkatan dari PHP : Hypertext Prepocessor. PHP digunakan sebagai salah satu script untuk membuat fungsi-fungsi sebagai mempermudah sistem teknis dalam website. Dalam praktiknya PHP biasanya digunakan bersama dengan penggunaan bahasa pemrograman lainnya seperti bahasa pemrograman HTML dan bahasa pemrograman JavaScript.

3.6. Laravel

Laravel adalah sebuah framework PHP yang dirilis dibawah lisensi MIT, dibangun dengan konsep MVC (model view controller). Laravel adalah pengembangan website berbasis MVP yang ditulis dalam PHP yang dirancang untuk meningkatkan kualitas perangkat lunak dengan mengurangi biaya pengembangan awal dan biaya pemeliharaan, dan untuk meningkatkan pengalaman bekerja dengan aplikasi dengan menyediakan sintaks yang ekspresif, jelas dan menghemat waktu.

3.7. ReactJS

ReactJS adalah sebuah pustaka/*library* javascript yang bersifat open source untuk membangun User Interface yang dibuat oleh Facebook. ReactJS hanya mengurusi semua hal yang berkaitan dengan tampilan dan logika di sekitarnya. React JS dapat mendesain tampilan sederhana untuk setiap level dalam aplikasi, sehingga dapat digunakan untuk membuat dan mengembangkan pembuatan aplikasi berbasis web.

3.8. MySQL

MySQL adalah sebuah database manajemen sistem (DBMS) populer yang memiliki fungsi sebagai Relational Database Manajemen System (RDBMS). Selain itu MySQL software merupakan suatu aplikasi yang sifatnya open source serta server basis data MySQL memiliki kinerja sangat cepat, reliable, dan mudah untuk digunakan serta bekerja dengan arsitektur client server atau embedded systems. Dikarenakan faktor open source dan populer tersebut maka cocok untuk mendemontrasikan proses replikasi basis data.

3.9. Visual Studio Code

Visual Studio Code adalah editor kode yang dikembangkan oleh Microsoft untuk Windows, Linux dan macOS. Visual Studio Code memberikan dukungan untuk debugging, kontrol Git tertanam, penyorotan sintaksis, penyelesaian kode cerdas,

snippet, dan refactoring kode. Visual Studio Code juga dapat disesuaikan, sehingga pengguna dapat mengubah tema editor, pintasan keyboard, dan preferensi.

[Halaman ini sengaja dikosongkan]

BAB IV
ANALISIS DAN DESAIN

4.1. Analisis Sistem

4.1.1. Definisi Umum Fitur

Terdapat 2 stakeholder utama dalam Aplikasi Web E-commerse SURAJ, yaitu user dan admin. User dapat melihat katalog produk serta melakukan pembelian pada produk-produk yang dikehendaki. Sedangkan Admin dapat mengelola produk serta pemesanan dan transaksi yang dilakukan oleh pengguna.

4.1.2. Analisis Kebutuhan Fungsional

Beberapa kebutuhan fungsional yang diperlukan pada Aplikasi Web E-commerce SURAJ dapat dilihat pada tabel 1.

Tabel 1: Kebutuhan fungsional

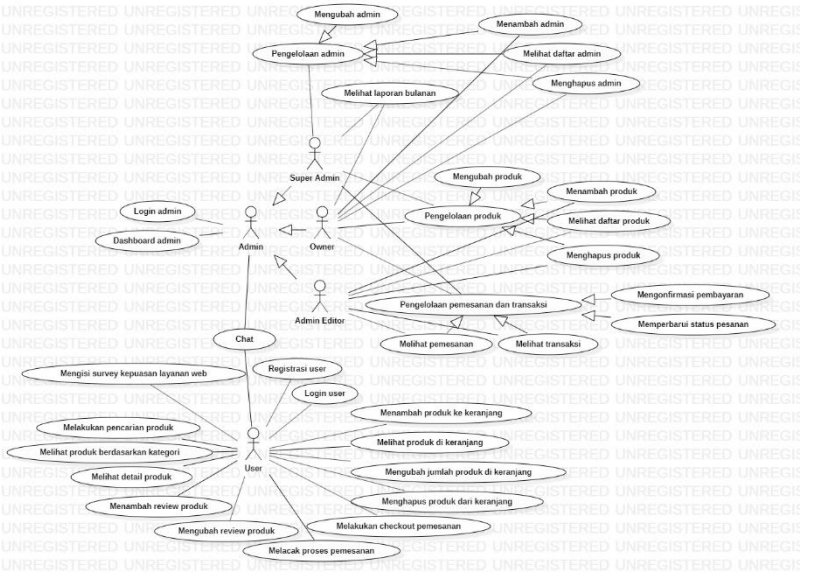
Kode Kebutuhan	Deskripsi Kebutuhan
FR-001	Melakukan login admin
FR-002	Melihat dashboard admin
FR-003	Menambah produk
FR-004	Melihat daftar produk
FR-005	Mengubah produk
FR-006	Menghapus produk
FR-007	Melihat pemesanan
FR-008	Melihat transaksi

FR-009	Mengonfirmasi pembayaran
FR-010	Memperbarui status pesanan
FR-011	Melihat laporan bulanan
FR-012	Menambah admin
FR-013	Melihat daftar admin
FR-014	Mengubah admin
FR-015	Menghapus admin
FR-016	Melakukan registrasi user
FR-017	Melakukan login user
FR-018	Melakukan pencarian produk
FR-019	Melihat produk berdasarkan kategori
FR-020	Melihat detail produk
FR-021	Melakukan chat
FR-022	Menambah produk ke keranjang
FR-023	Melihat produk di keranjang
FR-024	Mengubah jumlah produk di keranjang
FR-025	Menghapus produk dari keranjang
FR-026	Melakukan checkout pemesanan
FR-027	Melacak proses pemesanan
FR-028	Menambah review produk
FR-029	Mengubah review produk
FR-030	Mengisi survey kepuasan layanan web

4.2. Diagram Kasus Penggunaan

Pembahasan dengan pembimbing lapangan tentang fitur-fitur yang perlu ada dalam Aplikasi Web E-commerce SURAJ menghasilkan beberapa fitur yang dijadikan

diagram kasus penggunaan (Use Case Diagram) sehingga memudahkan untuk dipahami. Use Case Diagram yang telah dibuat dapat dilaht pada Gambar 4.1.



Gambar 1: Diagram Use Case Aplikasi Web E-commerce SURAJ

4.3. Spesifikasi Kasus Penggunaan

4.3.1. Login admin

Tabel 2: Spesifikasi Use Case login admin

Kode Use Case	UC001
Nama Use Case	Login admin
Aktor	Admin

Deskripsi	Proses untuk memasukkan admin ke dalam sistem
Kondisi Awal	Admin belum masuk ke dalam sistem
Kondisi Akhir	Admin masuk ke dalam sistem
Alur Normal	
Aktor	Sistem
1. Admin membuka halaman login untuk admin 3. Admin memasukkan username dan password 4. Admin menekan tombol login	2. Sistem menampilkan form untuk login 5. Sistem memeriksa data yang dimasukkan A.1. Data yang dimasukkan tidak sesuai 6. Sistem memasukkan admin ke dalam sistem
Alur Alternatif	
A.1. Data yang dimasukkan tidak sesuai	
Aktor	Sistem
2. Admin membaca pesan tersebut 3. Admin kembali ke alur normal nomor 3	1. Sistem menampilkan pesan error bahwa terdapat data yang dimasukkan tidak sesuai

4.3.2. Dashboard admin

Tabel 3: Spesifikasi Use Case dashboard admin

Kode Use Case	UC002
Nama Use Case	Dashboard admin
Aktor	Admin
Deskripsi	Proses untuk menampilkan ringkasan produk, pemesanan, dan transaksi penjualan
Kondisi Awal	Admin telah melakukan login
Kondisi Akhir	Halaman dashboard ditampilkan
Alur Normal	
Aktor	Sistem
1. Admin membuka halaman dashboard	2. Sistem menampilkan halaman dashboard

4.3.3. Menambah produk

Tabel 4: Spesifikasi Use Case menambah produk

Kode Use Case	UC003
Nama Use Case	Menambah produk
Aktor	Admin
Deskripsi	Proses untuk menambahkan produk baru ke dalam katalog
Kondisi Awal	Admin telah melakukan login
Kondisi Akhir	Produk berhasil ditambahkan

Alur Normal	
Aktor	Sistem
1. Admin membuka halaman untuk menambahkan produk	2. Sistem menampilkan form untuk menambahkan produk
3. Admin memasukkan data produk. Data produk meliputi: - Nama - Kategori - Stok - Harga - Deskripsi - Foto E.1. Admin menekan tombol kembali	4. Sistem memeriksa data yang dimasukkan A.1. Data yang dimasukkan tidak lengkap A.2. Ukuran file foto produk lebih dari 2MB 5. Sistem menyimpan data produk ke dalam database
Alur Alternatif	
A.1. Data yang dimasukkan tidak lengkap	
Aktor	Sistem

2. Admin membaca pesan tersebut 3. Admin kembali ke alur normal nomor 3	1. Sistem menampilkan pesan error bahwa terdapat data yang belum lengkap/belum diisi
A.2. Ukuran file foto produk yang diunggah melebihi 2MB	
Aktor	Sistem
2. Admin membaca pesan tersebut 3. Admin kembali ke alur normal nomor 3	1. Sistem menampilkan pesan error bahwa ukuran file terlalu besar
Alur Eksepsi	
E.1. Admin menekan tombol kembali	
Aktor	Sistem
	1. Sistem menampilkan halaman daftar produk

4.3.4. Melihat daftar produk

Tabel 5: Spesifikasi Use Case melihat daftar produk

Kode Use Case	UC004
Nama Use Case	Melihat daftar produk
Aktor	Admin
Deskripsi	Proses untuk melihat produk yang sudah ada dalam katalog

Kondisi Awal	Admin telah melakukan login
Kondisi Akhir	Admin dapat melihat daftar produk yang sudah ada dalam katalog
Alur Normal	
Aktor	Sistem
1. Admin membuka halaman untuk melihat daftar produk	2. Sistem menampilkan halaman daftar produk

4.3.5. Mengubah produk

Tabel 6: Spesifikasi Use Case mengubah produk

Kode Use Case	UC005
Nama Use Case	Mengubah produk
Aktor	Admin
Deskripsi	Proses untuk mengubah data dari produk yang sudah ada dalam katalog
Kondisi Awal	Admin telah melakukan login
Kondisi Akhir	Data produk berhasil diubah
Alur Normal	
Aktor	Sistem
1. Admin membuka halaman untuk melihat daftar produk	2. Sistem menampilkan

	halaman yang berisi daftar produk
3. Admin menekan tombol "ubah" pada produk yang ingin diubah datanya	4. Sistem menampilkan form untuk mengubah produk
5. Admin memasukkan data yang ingin diubah (semua data produk dapat diubah) E.1. Admin menekan tombol kembali	6. Sistem memeriksa data yang dimasukkan A.1. Data yang dimasukkan tidak lengkap A.2. Ukuran file foto produk lebih dari 2MB 7. Sistem menyimpan data produk ke dalam database
Alur Alternatif	
A.1. Data yang dimasukkan tidak lengkap	
Aktor	Sistem
2. Admin membaca pesan tersebut 3. Admin kembali ke alur normal nomor 3	1. Sistem menampilkan pesan error bahwa terdapat data yang belum lengkap/belum diisi
A.2. Ukuran file foto produk yang diunggah melebihi 2MB	
Aktor	Sistem

2. Admin membaca pesan tersebut 3. Admin kembali ke alur normal nomor 3	1. Sistem menampilkan pesan error bahwa ukuran file terlalu besar
Alur Eksepsi	
E.1. Admin menekan tombol kembali	
Aktor	Sistem
	1. Sistem menampilkan halaman daftar produk

4.3.6. Menghapus produk

Tabel 7: Spesifikasi Use Case menghapus produk

Kode Use Case	UC006
Nama Use Case	Menghapus produk
Aktor	Admin
Deskripsi	Proses untuk menghapus produk dari katalog
Kondisi Awal	Admin telah melakukan login
Kondisi Akhir	Produk berhasil dihapus
Alur Normal	
Aktor	Sistem
1. Admin membuka halaman untuk melihat daftar produk	2. Sistem menampilkan

	halaman yang berisi daftar produk
3. Admin menekan tombol "hapus" pada produk yang ingin dihapus A.1. Admin menekan tombol "ubah"	
	4. Sistem menampilkan pop-up berisi konfirmasi penghapusan produk
5. Admin menekan tombol konfirmasi E.1. Admin membatalkan penghapusan produk	
	6. Sistem menghapus data produk dari database
Alur Alternatif	
A.1. Admin menekan tombol "ubah"	
Aktor	Sistem
	1. Sistem menampilkan form untuk mengubah produk
2. Admin menekan tombol "hapus"	
	3. Sistem kembali ke alur normal nomor 4
Alur Eksepsi	
E.1. Admin membatalkan penghapusan produk	
Aktor	Sistem
	1. Sistem menampilkan halaman daftar produk

4.3.7. Melihat pemesanan

Tabel 8: Spesifikasi Use Case melihat pemesanan

Kode Use Case	UC007
Nama Use Case	Melihat pemesanan
Aktor	Admin
Deskripsi	Proses untuk melihat daftar pemesanan yang telah dilakukan oleh user
Kondisi Awal	Admin telah melakukan login
Kondisi Akhir	Admin dapat melihat daftar pemesanan
Alur Normal	
Aktor	Sistem
1. Admin membuka halaman untuk melihat daftar pemesanan	2. Sistem menampilkan halaman daftar pemesanan

4.3.8. Melihat transaksi

Tabel 9: Spesifikasi Use Case melihat transaksi

Kode Use Case	UC008
Nama Use Case	Melihat transaksi
Aktor	Admin
Deskripsi	Proses untuk melihat daftar transaksi yang telah dilakukan oleh user

Kondisi Awal	Admin telah melakukan login
Kondisi Akhir	Admin dapat melihat daftar transaksi
Alur Normal	
Aktor	Sistem
1. Admin membuka halaman untuk melihat daftar transaksi	2. Sistem menampilkan halaman daftar transaksi

4.3.9. Mengonfirmasi pembayaran

Tabel 10: Spesifikasi Use Case mengonfirmasi pembayaran

Kode Use Case	UC009
Nama Use Case	Mengonfirmasi pembayaran
Aktor	Admin
Deskripsi	Proses untuk melakukan konfirmasi pada pembayaran yang telah dilakukan oleh user
Kondisi Awal	Admin telah melakukan login
Kondisi Akhir	Pembayaran berhasil dikonfirmasi
Alur Normal	
Aktor	Sistem
1. Admin membuka halaman untuk melihat daftar transaksi	2. Sistem menampilkan halaman daftar transaksi

3. Admin menekan tombol "konfirmasi" pada transaksi yang ingin dikonfirmasi	4, Sistem memperbarui status pemesanan

4.3.10. Memperbarui status pesanan

Tabel 11: Spesifikasi Use Case memperbarui status pesanan

Kode Use Case	UC010
Nama Use Case	Memperbarui status pesanan
Aktor	Admin
Deskripsi	Proses untuk mengubah status pesanan supaya user dapat melakukan tracking
Kondisi Awal	Admin telah melakukan login
Kondisi Akhir	Status pesanan berhasil diubah
Alur Normal	
Aktor	Sistem
1. Admin membuka halaman untuk melihat daftar pemesanan 3. Admin memilih status pesanan pada salah satu pemesanan yang transaksinya sudah dikonfirmasi.	2. Sistem menampilkan halaman daftar pemesanan

Pilihan status pesanan meliputi: - Diproses - Dikirim - Selesai - Dibatalkan	4. Sistem menyimpan status pesanan ke dalam database

4.3.11. Melihat laporan bulanan

Tabel 12: Spesifikasi Use Case melihat laporan bulanan

Kode Use Case	UC011
Nama Use Case	Melihat Laporan Bulanan
Aktor	Admin
Deskripsi	Proses untuk menampilkan halaman laporan bulanan
Kondisi Awal	Admin telah login ke dalam sistem
Kondisi Akhir	Sistem menampilkan halaman laporan bulanan
Alur Normal	
Aktor	Sistem
1. Admin membuka halaman laporan bulanan	2. Sistem menampilkan halaman "Laporan"

3. Admin dapat memilih laporan bulanan yang ingin di lihat (di cek)	4. Sistem menampilkan laporan bulanan sesuai bulan yang di pilih oleh admin

4.3.12. Menambah admin

Tabel 13: Spesifikasi Use Case menambah admin

Kode Use Case	UC012
Nama Use Case	Menambah admin
Aktor	Super Admin
Deskripsi	Proses untuk menambahkan admin baru
Kondisi Awal	Super Admin telah melakukan login
Kondisi Akhir	Admin berhasil ditambahkan
Alur Normal	
Aktor	Sistem
1. Super Admin membuka halaman untuk menambahkan admin 3. Super Admin memasukkan data admin. Data admin meliputi:	2. Sistem menampilkan form untuk menambahkan admin

- Nama - Username - Email - Nomor HP - Peran (Super Admin/Admin) E.1. Super Admin menekan tombol kembali	4. Sistem memeriksa data yang dimasukkan A.1. Data yang dimasukkan tidak lengkap 5. Sistem menyimpan data produk ke dalam database
Alur Alternatif	
A.1. Data yang dimasukkan tidak lengkap	
Aktor	Sistem
2. Super Admin membaca pesan tersebut 3. Super Admin kembali ke alur normal nomor 3	1. Sistem menampilkan pesan error bahwa terdapat data yang belum lengkap/belum diisi
Alur Eksepsi	
E.1. Super Admin menekan tombol kembali	
Aktor	Sistem
	1. Sistem menampilkan halaman daftar admin

4.3.13. Melihat daftar admin

Tabel 14: Spesifikasi Use Case melihat daftar admin

Kode Use Case	UC013
Nama Use Case	Melihat daftar admin
Aktor	Super Admin
Deskripsi	Proses untuk melihat admin yang sudah terdaftar
Kondisi Awal	Super Admin telah melakukan login
Kondisi Akhir	Super Admin dapat melihat daftar admin yang sudah terdaftar
Alur Normal	
Aktor	Sistem
1. Super Admin membuka halaman untuk melihat daftar admin	2. Sistem menampilkan halaman daftar admin

4.3.14. Mengubah admin

Tabel 15: Spesifikasi Use Case mengubah admin

Kode Use Case	UC014
Nama Use Case	Mengubah admin
Aktor	Super Admin
Deskripsi	Proses untuk mengubah data dari admin

Kondisi Awal	Super Admin telah melakukan login
Kondisi Akhir	Data admin berhasil diubah
Alur Normal	
Aktor	Sistem
1. Super Admin membuka halaman untuk melihat daftar admin	
	2. Sistem menampilkan halaman yang berisi daftar admin
3. Super Admin menekan tombol "ubah" pada admin yang ingin diubah datanya	
	4. Sistem menampilkan form untuk mengubah admin
5. Super Admin memasukkan data yang ingin diubah (semua data admin dapat diubah) E.1. Super Admin menekan tombol kembali	
	6. Sistem memeriksa data yang dimasukkan A.1. Data yang dimasukkan tidak lengkap 7. Sistem menyimpan data admin ke dalam database
Alur Alternatif	
A.1. Data yang dimasukkan tidak lengkap	
Aktor	Sistem

2. Super Admin membaca pesan tersebut 3. Super Admin kembali ke alur normal nomor 3	1. Sistem menampilkan pesan error bahwa terdapat data yang belum lengkap/belum diisi
Alur Eksepsi	
E.1. Super Admin menekan tombol kembali	
Aktor	Sistem
	1. Sistem menampilkan halaman daftar admin

4.3.15. Menghapus admin

Tabel 16: Spesifikasi Use Case menghapus admin

Kode Use Case	UC015
Nama Use Case	Menghapus admin
Aktor	Super Admin
Deskripsi	Proses untuk menghapus admin
Kondisi Awal	Super Admin telah melakukan login
Kondisi Akhir	Admin berhasil dihapus
Alur Normal	
Aktor	Sistem
1. Super Admin membuka halaman untuk melihat daftar admin	2. Sistem menampilkan

	halaman yang berisi daftar admin
3. Super Admin menekan tombol "hapus" pada admin yang ingin dihapus	4. Sistem menampilkan pop-up berisi konfirmasi penghapusan admin
5. Super Admin menekan tombol konfirmasi	
E.1. Super Admin membatalkan penghapusan admin	6. Sistem menghapus data admin dari database
Alur Eksepsi	
E.1. Super Admin membatalkan penghapusan admin	
Aktor	Sistem
	1. Sistem menampilkan halaman daftar admin

4.3.16. Registrasi user

Tabel 17: Spesifikasi Use Case registrasi user

Kode Use Case	UC016
Nama Use Case	Registrasi user
Aktor	User
Deskripsi	Proses membuat akun untuk user agar dapat melakukan pembelian produk
Kondisi Awal	User belum memiliki akun

Kondisi Akhir	User berhasil mendaftarkan akun
Alur Normal	
Aktor	Sistem
1. User membuka halaman registrasi 3. User memasukkan data akun. Data akun meliputi: - Nama depan - Nama belakang - Email - No HP - Password	2. Sistem menampilkan form registrasi 4. Sistem memeriksa data yang dimasukkan A.1. Data yang dimasukkan tidak lengkap 5. Sistem menyimpan data akun ke dalam database
Alur Alternatif	
A.1. Data yang dimasukkan tidak lengkap	
Aktor	Sistem
2. User membaca pesan tersebut	1. Sistem menampilkan pesan error bahwa terdapat data yang belum lengkap/belum diisi

3. User kembali ke alur normal nomor 3	

4.3.17. Login user

Tabel 18: Spesifikasi Use Case login user

Kode Use Case	UC017
Nama Use Case	Login user
Aktor	User
Deskripsi	Proses untuk memasukkan user ke dalam sistem
Kondisi Awal	User belum masuk ke dalam sistem
Kondisi Akhir	User masuk ke dalam sistem
Alur Normal	
Aktor	Sistem
1. User membuka halaman login 3. Admin memasukkan email dan password 4. Admin menekan tombol login	2. Sistem menampilkan form untuk login 5. Sistem memeriksa data yang dimasukkan A.1. Data yang dimasukkan tidak sesuai 6. Sistem memasukkan user ke dalam sistem
Alur Alternatif	
A.1. Data yang dimasukkan tidak sesuai	

Aktor	Sistem
2. User membaca pesan tersebut 3. User kembali ke alur normal nomor 3	1. Sistem menampilkan pesan error bahwa terdapat data yang dimasukkan tidak sesuai

4.3.18. Melakukan pencarian produk

Tabel 19: Spesifikasi Use Case melakukan pencarian produk

Kode Use Case	UC018
Nama Use Case	Melakukan pencarian produk
Aktor	User
Deskripsi	Proses untuk pencarian produk menggunakan fitur search
Kondisi Awal	-
Kondisi Akhir	Sistem hanya menampilkan produk berdasarkan hasil pencarian yang dilakukan *user*
Alur Normal	
Aktor	Sistem
1. *User* memasukkan nama produk yang ingin di cari pada fitur search	2. Sistem menampilkan produk yang sesuai dengan yang telah dicari oleh *user*
Alur Alternatif	

A.1. Data yang dimasukkan tidak ada	
Aktor	Sistem
1. *User* memasukkan nama produk yang ingin di cari	2. Sistem menampilkan teks bahwa produk yang dicari tidak tersedia

4.3.19. Menampilkan produk berdasarkan kategori

Tabel 20: Spesifikasi Use Case menampilkan produk berdasarkan kategori

Kode Use Case	UC019
Nama Use Case	Menampilkan produk berdasarkan kategori
Aktor	User
Deskripsi	Proses menampilkan produk berdasarkan kategori yang telah dipilih oleh *user*
Kondisi Awal	-
Kondisi Akhir	Sistem hanya menampilkan daftar produk berdasarkan kategori yang telah dipilih oleh *user*
Alur Normal	
Aktor	Sistem
1. *User* menekan tombol kategori pada navigasi bar,	

lalu memilih nama kategori produk yang ingin ditampilkan	2. Sistem menampilkan daftar produk yang sesuai dengan kategori yang dipilih oleh *user*

4.3.20. Melihat detail produk

Tabel 21: Spesifikasi Use Case melihat detail produk

Kode Use Case	UC020
Nama Use Case	Melihat detail produk
Aktor	User
Deskripsi	Proses untuk melihat detail produk yang dipilih oleh *user*
Kondisi Awal	-
Kondisi Akhir	Sistem menampilkan detail dari produk yang telah dipilih oleh aktor
Alur Normal	
Aktor	Sistem
2. *User* memilih produk yang ingin dilihat detailnya	1. Sistem menampilkan produk 3. Sistem menampilkan detail dari produk yang telah dipilih oleh *user*

4.3.21. Chat

Tabel 22: Spesifikasi Use Case chat

Kode Use Case	UC021
Nama Use Case	Chat
Aktor	User
Deskripsi	Proses untuk aktor melakukan konsultasi dengan dokter yang tersedia dengan fitur *chat*
Kondisi Awal	*user* melakukan login
Kondisi Akhir	*user* berhasil menggunakan fitur *chat* untuk konsultasi dengan dokter
Alur Normal	
Aktor	Sistem
1. *user* membuka fitur *chat* 3. *user* dapat berkonsultasi dengan dokter melalui fitur *chat*	2. Sistem menampilkan fitur *chat* 4. Sistem akan menampilkan balasan dari dokter

4.3.22. Menambah produk ke keranjang

Tabel 23: Spesifikasi Use Case menambah produk ke keranjang

Kode Use Case	UC022
Nama Use Case	Menambah produk ke keranjang
Aktor	User

Deskripsi	Proses untuk menambahkan produk yang akan dibeli ke keranjang
Kondisi Awal	User telah melakukan login
Kondisi Akhir	Produk telah ditambahkan ke keranjang
Alur Normal	
Aktor	Sistem
1. User membuka halaman untuk melihat detail produk 3. User mengisi jumlah produk yang diinginkan 4. User menekan tombol "tambah ke keranjang"	2. Sistem menampilkan halaman yang berisi detail produk 5. Sistem menambahkan produk ke keranjang

4.3.23. Melihat produk di keranjang

Tabel 24: Spesifikasi Use Case melihat produk di keranjang

Kode Use Case	UC023
Nama Use Case	Melihat produk di keranjang
Aktor	User
Deskripsi	Proses untuk melihat produk yang sudah ada dalam keranjang
Kondisi Awal	User telah melakukan login

Kondisi Akhir	User dapat melihat produk yang akan dibeli di keranjang
Alur Normal	
Aktor	Sistem
1. User membuka halaman untuk melihat produk di keranjang	2. Sistem menampilkan halaman keranjang

4.3.24. Mengubah jumlah produk di keranjang

Tabel 25: Spesifikasi Use Case mengubah jumlah produk di keranjang

Kode Use Case	UC024
Nama Use Case	Mengubah jumlah produk di keranjang
Aktor	User
Deskripsi	Proses untuk mengubah jumlah produk yang sudah ada dalam keranjang
Kondisi Awal	User telah melakukan login
Kondisi Akhir	Jumlah produk berhasil diubah
Alur Normal	
Aktor	Sistem
1. User membuka halaman keranjang	2. Sistem menampilkan halaman keranjang

3. User memasukkan nilai pada produk yang ingin diubah jumlahnya	4. Sistem menyimpan jumlah produk di keranjang

4.3.25. Menghapus produk dari keranjang

Tabel 26: Spesifikasi Use Case menghapus produk dari keranjang

Kode Use Case	UC025
Nama Use Case	Menghapus produk dari keranjang
Aktor	User
Deskripsi	Proses untuk menghapus produk dari keranjang
Kondisi Awal	User telah melakukan login
Kondisi Akhir	Produk berhasil dihapus dari keranjang
Alur Normal	
Aktor	Sistem
1. User membuka halaman keranjang 3. User menekan tombol "hapus" pada produk yang ingin dihapus	2. Sistem menampilkan halaman keranjang 4. Sistem menghapus produk dari keranjang

4.3.26. Melakukan checkout pemesanan

Tabel 27: Spesifikasi Use Case melakukan checkout pemesanan

Kode Use Case	UC026
Nama Use Case	Melakukan checkout pemesanan
Aktor	User
Deskripsi	Proses untuk melakukan transaksi pemesanan pada produk yang hendak dibeli
Kondisi Awal	User telah melakukan login
Kondisi Akhir	User telah melakukan transaksi pemesanan
Alur Normal	
Aktor	Sistem
1. User membuka halaman keranjang A.1. User menekan tombol "beli sekarang" di halaman detail produk 3. User memilih produk yang hendak dibeli 5. User menekan tombol lanjut 7. User memasukkan data pengiriman dan pembayaran.	 2. Sistem menampilkan halaman keranjang 4. Sistem menampilkan subtotal pembayaran 6. Sistem menampilkan halaman checkout

Data pengiriman dan pembayaran meliputi: - Nama depan - Nama belakang - Alamat lengkap - Kecamatan - Kota - Provinsi - Kode pos - Negara - No HP - Metode - Nominal	
E.1. User menekan tombol kembali	8. Sistem menambahkan data pengiriman dan pembayaran ke dalam database
Alur Alternatif	
A.1. User menekan tombol "beli sekarang" di halaman detail produk	
Aktor	Sistem
2. User kembali ke alur normal nomor 7	1. Sistem menampilkan halaman halaman checkout
Alur Eksepsi	
E.1. User menekan tombol kembali	
Aktor	Sistem

	1. Sistem menampilkan halaman keranjang

4.3.27. Melacak proses pemesanan

Tabel 28: Spesifikasi Use Case melacak proses pemesanan

Kode Use Case	UC027
Nama Use Case	Melacak proses pemesanan
Aktor	User
Deskripsi	Proses untuk melacak proses pemesanan atau lokasi (posisi) barang yang telah dipesan oleh *user*
Kondisi Awal	*user* telah *login* dan melakukan pemesanan
Kondisi Akhir	*user* mengetahui proses pemesanan yang terjadi atau posisi (lokasi) barang yang dipesan
Alur Normal	
Aktor	Sistem
1. *User* membuka halaman untuk *track* pemesanan	2. Sistem menampilkan halaman *track* pemesanan

4.3.28. Menambah review produk

Tabel 29: Spesifikasi Use Case menambah review produk

Kode Use Case	UC028
Nama Use Case	Menambah review produk

Aktor	User
Deskripsi	Proses untuk menambah testimoni mengenai produk yang sudah dibeli
Kondisi Awal	User telah selesai melakukan pembelian produk
Kondisi Akhir	Review produk berhasil ditambahkan
Alur Normal	
Aktor	Sistem
1. User membuka halaman dari produk yang sudah dibeli 3. User memasukkan rating dan komentar	2. Sistem menampilkan halaman dari produk yang sudah dibeli 4. Sistem menyimpan review pengguna ke dalam database

4.3.29. Mengubah review produk

Tabel 30: Spesifikasi Use Case mengubah review produk

Kode Use Case	UC029
Nama Use Case	Mengubah review produk
Aktor	User
Deskripsi	Proses untuk mengubah testimoni yang sudah ada mengenai produk yang sudah dibeli

Kondisi Awal	User telah menambahkan review produk
Kondisi Akhir	Review produk berhasil diubah
Alur Normal	
Aktor	Sistem
1. User membuka halaman dari produk yang sudah dibeli	2. Sistem menampilkan halaman dari produk yang sudah dibeli
3. User mengubah rating dan/atau komentar	4. Sistem menyimpan review pengguna ke dalam database

4.3.30. Mengisi survey kepuasan layanan web

Tabel 31: Spesifikasi Use Case mengisi survey kepuasan layanan web

Kode Use Case	UC030
Nama Use Case	Mengisi survey kepuasan layanan web
Aktor	User
Deskripsi	Proses menambahkan komentar pengguna terhadap layanan web ke dalam database
Kondisi Awal	User telah melakukan konfirmasi bahwa pesanan telah sampai tujuan

Kondisi Akhir	Survey berhasil diisi
Alur Normal	
Aktor	Sistem
1. User mengonfirmasi bahwa pesanan telah sampai 3. User memasukkan komentar	2. Sistem menampilkan pop-up berisi form survey kepuasan layanan web 4. Sistem menyimpan komentar user ke dalam database

[Halaman ini sengaja dikosongkan]

BAB V
IMPLEMENTASI SISTEM

Bab ini membahas tentang implementasi dari sistem yang kami buat. Implementasi ini akan dibagi ke dalam beberapa bagian, yaitu bagian implementasi lapisan kontrol dan implementasi antarmuka pengguna.

5.1.Implementasi Lapisan Kontrol

Implementasi lapisan kontrol ini berisi logika yang digunakan aplikasi seperti kontrol untuk memasukkan data ke databases.

5.1.1. Cart Controller

Lapisan ini bertugas untuk menghubungkan form Keranjang dengan database agar data yang telah diisikan pada form dapat tersimpan di database.

```
<?php

namespace App\Http\Controllers;

use App\Cart;
use App\Product;
use
Illuminate\Database\Eloquent\ModelNotFoundExcepti
on;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Validator;
```

```
class CartController extends Controller
{
    public function index()
    {
        $user_id = auth('api')->user()->getKey();
        $cartItems = Cart::where('user_id',
$user_id)->get();

        return response([
            'success' => true,
            'message' => 'cart items retrieved
successfully.',
            'data' => $cartItems
        ]);
    }

    public function store(Request $request)
    {
        $validator = Validator::make($request-
>all(), [
            'quantity' => 'required|int',
            'note' => 'required|string',
        ]);
        if ($request->fails()) {
            return response([
                'success' => false,
                'message' => 'something went
wrong',
                'error_message' => $validator-
>messages()
            ]);
        }

        $user_id = auth('api')->user()->getKey();
```

```
        $product_id = $request-
>input('product_id');
        $quantity = $request->input('quantity');

        // Check if product exist or return 404
not found.
        try {
            Product::findOrFail($product_id);
        } catch (ModelNotFoundException $e) {
            return response([
                'success' => false,
                'message' => 'The Product you\'re
trying to add does not exist.',
            ]);
        }

        // Check if the same product is already
in the Cart, if true update the quantity, if not
create new one
        $cartItem = Cart::where(['user_id' =>
$user_id, 'product_id' => $product_id])->first();

        if ($cartItem) {
            $cartItem->quantity = $quantity;
            $cartItem = Cart::where([
                'user_id' => $user_id,
                'product_id' => $product_id])-
>update(['quantity' => $quantity]);

            return response([
                'success' => true,
                'message' => 'The Cart was
updated with the given product information
successfully',
                'data' => $cart_item
```

```
            ]);
        } else {
            $cart_item = Cart::create([
                'user_id' => $user_item,
                'product_id' =>
$request['product_id'],
                'quantity' =>
$request['quantity'],
                'note' => $request['note']
            ]);

            return response([
                'success' => true,
                'message' => 'Product
successfully added to cart.',
                'data' => $cart_item,
            ]);
        }
    }

    public function update($id, Request $request)
    {
        try {
            $item = Cart::find($id);
            $item->quantity = $item ['quantity'];
            $item->note = $request['note'];
            $item->save();
        } catch (\Exception $exception) {
            return response([
                'success' => false,
                'message' => 'something went
wrong',
                'error_message' => $exception-
>getMessage(),
            ]);
```

```
        }

        return response([
            'success' => true,
            'message' => 'cart item successfully
updated.',
            'data' => $item
        ]);
    }

    public function destroy($id)
    {
        try {
            $item = Cart::where('id', $id)-
>delete();
        } catch (\Exception $exception) {
            return response([
                'success' => false,
                'message' => 'something went
wrong',
                'error_message' => $exception-
>getMessage(),
            ]);
        }

        return response([
            'success' => true,
            'message' => 'delete item in cart
success.',
            'data' => $item,
        ]);
    }
}
```

5.1.2. Category Controller

Lapisan ini bertugas untuk menghubungkan kategori produk dengan database agar data yang telah diisikan pada form dapat tersimpan di database.

```
<?php

namespace App\Http\Controllers;

use App\Category;
use
Illuminate\Database\Eloquent\ModelNotFoundExcepti
on;
use Illuminate\Http\Request;

class CategoryController extends Controller
{
    public function store(Request $request)
    {
        $request->validate(['name' =>
'required']);

        try {
            $category = Category::create([
                'name' => $request['name'],
            ]);
        } catch (ModelNotFoundException $e) {
            return response([
                'success' => false,
                'message' => 'The Product you\'re
trying to add does not exist.',
            ]);
        }
```

```
        return response([
            'success' => true,
            'message' => 'cart items created
successfully.',
            'data' => $category
        ]);
    }

    public function index()
    {
        $categories = Category::all();
        return response([
            'success' => true,
            'message' => 'categories retrieved
successfully.',
            'data' => $categories
        ]);
    }

    public function show($id)
    {
        $category = Category::where('id', $id)-
>first();

        return response([
            'success' => true,
            'message' => 'category found
successfully.',
            'data' => $category
        ]);
    }

    public function update(Request $request, $id)
    {
```

```
        try {
            $category = Category::find($id);
            $category->name = $request['name'];
            $category->save();
        } catch (\Exception $exception) {
            return response([
                'success' => false,
                'message' => 'something went
wrong',
                'error_message' => $exception-
>getMessage(),
            ]);
        }

        return response([
            'success' => true,
            'message' => 'update category
success.',
            'data' => $category,
        ]);
    }

    public function destroy($id)
    {
        try {
            $category = Category::where('id',
$id)->delete();
        } catch (\Exception $exception) {
            return response([
                'success' => false,
                'message' => 'something went
wrong',
                'error_message' => $exception-
>getMessage(),
            ]);
```

```
        }

        return response([
            'success' => true,
            'message' => 'delete category
success.',
            'data' => $category,
        ]);
    }
}
```

5.1.3. Chat Controller

Lapisan ini bertugas untuk menghubungkan fitur Chat dengan database agar data yang telah dikirim dapat tersimpan di database.

```
<?php

namespace App\Http\Controllers;

use App\Chat;
use Illuminate\Http\Request;

class ChatController extends Controller
{
    public function __construct()
    {
        $this->clean();
    }

    private function clean()
    {
```

```
        return Chat::where('created_at', '<',
'NOW() - INTERVAL 180 DAY')->delete();
    }

    Public function send(Request $request)
    {
        try {
            $chat = Chat::create([
                'receiver_id' =>
$request['receiver_id'],
                'sender_id' =>
$request['sender_id'],
                'message' => $request['message'],
            ]);
        } catch (\Exception $exception) {
            return response([
                'success' => false,
                'message' => 'something went
wrong',
                'error_message' => $exception-
>getMessage(),
            ], 500);
        }

        return response([
            'success' => true,
            'message' => 'message sent
successfully',
            'data' => $chat
        ]);
    }

    Public function fetch($sender_id,
$receiver_id)
    {
```

```
        $messages = Chat::where('sender_id',
$sender_id)->orWhere('sender_id', $receiver_id)-
>get();

        return response([
            'success' => true,
            'message' => 'messages fetched',
            'data' => $messages
        ]);
    }
}
```

5.1.4. Payment Controller

Lapisan ini bertugas untuk menghubungkan fitur Pembayaran dengan database agar data yang telah dikirim dapat tersimpan di database.

```
<?php

namespace App\Http\Controllers;

use App\Http\Traits\TransactionTrait;
use App\Payment;


class PaymentController extends Controller
{
    use TransactionTrait;

    public function store($transaction_id)
    {
        $payment = Payment::create([
            'transaction_id' => $transaction_id,
            'is_verified' => 0
```

```
        ]);

        return response([
            'success' => true,
            'message' => 'payment created
successfully.',
            'data' => $payment
        ]);
    }

    public function index()
    {
        $payments = Payment::all();

        return response([
            'success' => true,
            'message' => 'payment found(s).',
            'data' => $payments
        ]);
    }

    public function show($payments)
    {
        $payment = Payment::find($id);

        return response([
            'success' => true,
            'message' => 'payment found.',
            'data' => $payment
        ]);
    }

    public function updateVerify($id, $verify)
    {
```

```
        $payment = Payment::where('id', $id)-
>first();
        $payment->is_verified = $verify;

        //if payment verified, update transaction
status
        if ($payment->is_verified == 1) {
            $this->transactionUpdateStatus($this-
>transaction_id, 2);
        }
        $this->save();

        return response([
            'success' => true,
            'message' => 'payment verified
successfully.',
            'data' => $payment
        ]);
    }
}
```

5.1.5. Product Controller

Lapisan ini bertugas untuk menghubungkan Produk dengan database agar data yang telah dikirimkan melalui form dapat tersimpan di database.

```
<?php

namespace App\Http\Controllers;

use App\Category;
```

```
use App\Product;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Validator;

class ProductController extends Controller
{
    public function store(Request $request)
    {
        $validator = Validator::make($request-
>all(), [
            'name' => 'required',
            'category_id' => 'required|numeric',
            'quantity' => 'required|numeric',
            'description' => 'required',
            'price' => 'required|numeric',
            'length' => 'numeric',
            'width' => 'numeric',
            'weight' => 'numeric',
            'image' =>
'mimes:jpeg,png,jpg|max:10024',
        ]);

        if ($validator->fails()) {
            return response([
                'success' => false,
                'message' => 'something went
wrong',
                'error_message' => $validator-
>messages()
            ]);
        }

        try {
            $imgPath = null;
            if ($request->hasFile('image')) {
```

```
                $img = $request->file('image');
                $ext = $img-
>getClientOriginalExtension();
                $imgName = time() . str_replace('
', '', $request['name']) . '.' . $ext;
                $imgPath = $img-
>storeAs('public/product/' . str_replace(' ', '',
$request['categori-id']), $imgName);
            }

            $product = Product::create([
                'name' => $request['name'],
                'category_id' =>
$request['category_id'],
                'quantity' =>
$request['quantity'],
                'description' =>
$request['description'],
                'price' => $request['price'],
                'length' => $request['length'],
                'width' => $request['width'],
                'weight' => $request['weight'],
                'image' => $imgPath,
            ]);
        } catch (\Exception $exception) {
            return response([
                'success' => false,
                'message' => 'something went
wrong',
                'error_message' => $exception-
>getMessage(),
            ]);
        }

        return response([
```

```
            'success' => true,
            'message' => 'product stored
successfully',
            'data' => $product
        ]);
    }

    public function show($id)
    {
        $product = Product::find($id);

    }

    public function getByCategory($category)
    {
        $products = Product::where('category_id',
$category)->get();

        if (count($products) == 0) {
            return response([
                'success' => true,
                'message' => 'product(s) not
found',
                'data' => $products
            ]);
        }

        return response([
            'success' => true,
            'message' => 'product(s) found',
            'data' => $products
        ]);
    }
```

```
    public function search($query)
    {
        $products = Product::where('name',
'like', '%' . $query . '%')->get();

        if (count($products) == 0) {
            return response([
                'success' => true,
                'message' => 'product(s) not
found',
                'data' => $products
            ]);
        }

        return response([
            'success' => true,
            'message' => 'product(s) found',
            'data' => $products
        ]);
    }

    public function fetch()
    {
        $products = Product::all();

        return response([
            'success' => true,
            'message' => 'products fetched
successfully',
            'data' => $products
        ]);
    }

    public function update(Request $request,
$product)
```

```
    {
        try {
            $imgPath = $product->image;
            if ($request->hasFile('image')) {
                $img = $request->file('image');
                $ext = $img-
>getClientOriginalExtension();
                $imgName = time() . str_replace('
', '', $request['name'] . '.' . $ext);
                $imgPath = $img-
>storeAs('public/product' . str_replace(' ', '',
$request['category-id']), $imgName);
            }

            $product->name = $request['name'];
            $product->category_id =
$request['category-id'];
            $product->quantity =
$product['quantity'];
            $product->description =
$request['description'];
            $product->price = $request['price'];
            $product->length =
$request['length'];
            $product->width = $request['width'];
            $product->weight =
$request['weight'];
            $product->image = $imgPath;
            $product->save();
        } catch (\Exception $exception) {
            return response([
                'success' => false,
                'message' => 'something went
wrong',
```

```
                'error_message' => $exception-
>getMessage(),
            ]);
        }

        return response([
            'success' => true,
            'message' => 'product updated
successfully',
            'data' => $product
        ]);
    }

    public function destroy(Product $product)
    {
        try {
            $product = $product ->delete();
        } catch (\Exception $exception) {
            return response([
                'success' => false,
                'message' => 'something went
wrong',
                'error_message' => $exception-
>getMessage(),
            ]);
        }

        return response([
            'success' => true,
            'message' => 'product deleted
successfully',
            'data' => $product
        ]);
    }
```

```
    public function getReviews(Product $product)
    {
        // dd($product);
        $id = $product->id;
        $product = Product::where('id', $id)-
>first();
        $reviews = $product->reviews;
        // return response(['Reviews' =>
$reviews, 'message' => 'Retrieved successfully'],
200);
        return response([
            'success' => true,
            'message' => 'Retrieved
successfully',
            'data' => $reviews
        ]);
    }
}
```

5.1.6. Review Controller

Lapisan ini bertugas untuk menghubungkan Review dengan database agar data yang telah dikirimkan pada form dapat tersimpan di database.

```
<?php

namespace App\Http\Controllers;

use App\Review;
use App\Product;
use App\User;
```

```
use Illuminate\Http\Request;

class ReviewController extends Controller
{
    public function store(Request $request,
Product $product)
    {
        try {
            $request->validate([
                'comment' => 'required',
                'score' => 'required',
            ]);
            $user_id = auth('api')->user()-
>getKey();

            $review = Review::create([
                'user_id' => $user_id,
                'product_id' => $product->id,
                'comment' => $request['comment'],
                'score' => $request['score']
            ]);
        }catch (\Exception $exception){
            return response([
                'success' => false,
                'message' => 'something went
wrong',
                'error_code' => $exception-
>getCode(),
                'error_message' => $exception-
>getMessage()
            ]);
        }

        return response([
            'success' => true,
```

```
            'message' => 'review added
successfully',
            'data' => $review
        ]);
    }

    public function getById(Review $review)
    {
        return response([
            'success' => true,
            'message' => 'review found',
            'data' => $review
        ]);
    }

    public function destroy(Review $review)
    {
        //
    }
}
```

5.1.7. RoleManagement Controller

Lapisan ini bertugas untuk menghubungkan Role Management dengan database agar data yang telah dikirimkan pada form dapat tersimpan di database.

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Validator;
use Spatie\Permission\Models\Role;
```

```
use Spatie\Permission\Models\Permission;

use App\User;

class RoleManagementController extends Controller
{
    //
    public function __construct()
    {
        $this->middleware(['auth:api',
'role:super-admin']);
    }

    public function index()
    {
        $all_users_with_all_their_roles =
User::with('roles')->get();

        return response([
            'success' => true,
            'message' => 'Retrieved
successfully.',
            'data' =>
$all_users_with_all_their_roles
        ]);
    }

    public function update(Request $request)
    {
        $validator = Validator::make($request-
>all(), [
            'user_id' => 'required',
            'role_id' => 'required'
        ]);
```

```
        if ($validator->fails()) {
            return response([
                'success' => false,
                'message' => 'something went
wrong',
                'error_message' => $validator-
>errors()
            ]);
        }

        $user = User::find($request->user_id);
        $role = Role::find($request->role_id);

        $user->assignRole($role);

        return response([
            'success' => true,
            'message' => 'Role has been assigned
successfully.',
            'data' => $user
        ]);
    }

    public function remove(Request $request)
    {
        $validator = Validator::make($request-
>all(), [
            'user_id' => 'required',
            'role_id' => 'required'
        ]);

        if ($validator->fails()) {
            return response([
                'success' => false,
```

```
                'message' => 'something went
wrong',
                'error_message' => $validator-
>errors()
            ]);
        }

        $user = User::find($request->user_id);
        $role = Role::find($request->role_id);

        $user->removeRole($role);

        return response([
            'success' => true,
            'message' => 'Role has been removed
successfully.',
            'data' => $user
        ]);
    }
}
```

5.1.8. Shipment Controller

Lapisan ini bertugas untuk menghubungkan Shipment dengan database agar data yang telah dikirimkan dapat tersimpan di database.

```
<?php

namespace App\Http\Controllers;

use App\Http\Traits\TransactionTrait;
use App\Shipment;
```

```
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Validator;

class ShipmentController extends Controller
{
    use TransactionTrait;

    public function store(Request $request)
    {
        $validator = Validator::make($request-
>all(), [
            'transaction_id' =>
'required|numeric',
            'country_id' => 'required|numeric',
            'city' => 'required',
            'address' => 'required',
            'shipping_company' => 'required',
            'shipping_site' => 'required',
            'tracking_number' => 'required',
            'status' => 'required',
        ]);

        if ($validator->fails()) {
            return response([
                'success' => false,
                'message' => 'something went
wrong',
                'error_message' => $validator-
>messages()
            ]);
        }

        try {
            $shipping = Shipment::create([
```

```
                'transaction_id' =>
$request['transaction_id'],
                'country_id' =>
$request['country_id'],
                'city' => $request['city'],
                'address' => $request['address'],
                'shipping_company' =>
$request['shipping_company'],
                'shipping_site' =>
$request['shipping_site'],
                'tracking_number' =>
$request['tracking_number'],
                'status' => $request ['status'],
            ]);
            $this-
>transactionUpdateStatus($request['transaction_id
'], 4);
        } catch (\Exception $exception) {
            return response([
                'success' => false,
                'message' => 'something went
wrong',
                'error_message' => $exception-
>getMessage(),
            ]);
        }

        return response([
            'success' => true,
            'message' => 'shipping stored
successfully',
            'data' => $shipping
        ]);
    }
```

```
    public function updateStatus($shipping,
Request $request)
    {
        $validator = Validator::make($request-
>all(), [
            'status' => 'required',
        ]);

        if ($validator->fails()) {
            return response([
                'success' => false,
                'message' => 'something went
wrong',
                'error_message' => $validator-
>messages()
            ]);
        }
        try {
            $shipment = Shipment::where('id',
$id)->first();
            $shipment->status =
$request['status'];
            $shipment->save();
        } catch (\Exception $exception) {
            return response([
                'success' => false,
                'message' => 'something went
wrong',
                'error_message' => $exception-
>getMessage(),
            ]);
        }

        return response([
            'success' => true,
```

```
            'message' => 'shipping stored
successfully',
            'data' => $shipment
        ]);
    }

    public function index()
    {
        $shipments = Shipment::all();

        return response([
            'success' => true,
            'message' => 'shipment(s) found',
            'data' => $shipments
        ]);
    }

    public function show($id)
    {
        $shipment = Shipment::find($id);

        if (!$shipment) {
            return response([
                'success' => true,
                'message' => 'shipment not
found',
                'data' => $shipment
            ]);
        }

        return response([
            'success' => true,
            'message' => 'shipment found',
            'data' => $shipment
        ]);
```

```
    }
}
```

5.1.9. Transaction Controller

Lapisan ini bertugas untuk menghubungkan Transaction dengan database agar data yang telah dikirimkan dapat tersimpan di database.

```
<?php

namespace App\Http\Controllers;

use App\Cart;
use App\Product;
use App\Transaction;
use App\TransactionDetail;
use Carbon\Carbon;
use Illuminate\Http\Request;


class TransactionController extends Controller
{
    public function checkout(Request $request)
    {
        $items =
\GuzzleHttp\json_decode($request['items']);
        $user_id = auth('api')->user()->getKey();
        $total_price = 0;
        $item_quantity = 0;
        foreach ($items as $item) {
            $cartItem = Cart::find($item);
```

```
            $product = Product::where('id',
$cartItem->product_id)->first();
            $product_price = $product->price;
            $item_quantity = $cartItem->quantity;
            $sub_total = $product_price *
$item_quantity;
            $total_price = $total_price +
$sub_total;
        }

        $transaction = $this-
>store($total_price);

        foreach ($items as $item) {
            $cartItem = Cart::find($item);
            $product = Product::where('id',
$cartItem->product_id)->first();
            TransactionDetail::create([
                'transaction_id' => $transaction-
>id,
                'product_id' => $product->id,
                'quantity' => $item_quantity,
                'price' => $product->price,
            ]);

            // update product stock
            $product_quantity = $product-
>quantity;
            $new_product_quantity =
$product_quantity - $item_quantity;
            Product::where('id', $product->id)-
>update(['quantity' => $new_product_quantity]);
        }
```

```
        Cart::where('user_id', $user_id)-
>delete();

        return response([
            'success' => true,
            'message' => 'transaction created
successfully',
            'data' => $transaction
        ]);
    }

    private function store($total_price)
    {

        $total_price = auth('api')->user()-
>getKey();
        $now = Carbon::now()->toDateTimeString();

        $transaction = Transaction::create([
            'user_id' => $user_id,
            'date' => $now,
            'total' => $total_price,
            'status' => "1"
        ]);

        return $transaction;
    }

    public function show($id){
        $transaction = Transaction::find($id);
        $transaction =
TransactionDetail::where('transaction_id',
$transaction ->id)->get();
        return response([
            'success' => true,
```

```
            'message' => 'transaction created
successfully',
            'data' => $transaction,
            'data1' => $transaction
        ]);
    }
}
```

5.1.10. UserProfile Controller

Lapisan ini bertugas untuk menghubungkan User Profile dengan database agar data yang telah diisikan dapat tersimpan di database.

```
<?php

namespace App\Http\Controllers;

use App\User;
use App\UserProfile;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Validator;
use Illuminate\Support\MessageBag;


class UserProfileController extends Controller
{
    public function store(Request $request, $id)
    {
        $validator = Validator::make($validator-
>all(), [
            'name' => 'required|string',
```

```
            'avatar' =>
'mimes:jpeg,png,jpg|max:10024',
            'province' => 'required|string',
            'city' => 'required|string',
            'subdistrict' => 'required|string',
            'address' => 'required|string',
            'postal_code' => 'required',
            'country_id' => 'required|int',
            'phone' => 'required',
        ]);

        if ($validator->fails()) {
            return response([
                'success' => false,
                'message' => 'Something went
wrong.',
                'error_message' => $validator-
>errors()
            ]);
        }

        try {
            $imgPath = null;
            if  ($request->hasFile('avatar')) {
                $img = $request->file('avatar');
                $ext = $img-
>getClientOriginalExtension();
                $imgName = time() . str_replace('
', '', $request['name']) . '.' . $ext;
                $imgPath = $img-
>storeAs('public/user/' . str_replace(' ', '',
$id), $imgName);
            }

            $user_profile = UserProfile::create([
```

```
                'user_id' => $id,
                'name' => $request['name'],
                'province' =>
$request['province'],
                'city' => $request['city'],
                'subdistrict' =>
$request['subdistrict'],
                'address' => $request['address'],
                'postal_code' =>
$request['postal_code'],
                'country_id' =>
$request['country_id'],
                'phone' => $request['phone'],
                'avatar' => $imgPath,
            ]);
        } catch (\Exception $request) {
            return response([
                'success' => false,
                'message' => 'Something went
wrong',
                'error_message' => $exception-
>getMessage(),
            ]);
        }

        return response([
            'success' => true,
            'message' => 'User profile stored
successfully',
            'data' => $user_profile,
        ]);
    }

    public function update(Request $request, $id)
    {
```

```
        $user_profile =
UserProfile::where('user_id', $id)->first();

        try {
            $imgPath = $user_profile->avatar;
            if($request->hasFile('avatar')) {
                $img = $request->file('avatar');
                $ext = $img-
>getClientOriginalExtension();
                $imgName = time() . str_replace('
', '', $request['name']) . '.' . $ext;
                $imgPath = $img-
>storeAs('public/user/' . str_replace(' ', '',
$user_profile->user_id), $imgName);
            }

            $user_profile->name =
$request['name'];
            $user_profile->province =
$request['province'];
            $user_profile->city =
$request['city'];
            $user_profile->subdistrict =
$request['subdistrict'];
            $user_profile->address =
$request['address'];
            $user_profile->postal_code =
$request['postal_code'];
            $user_profile->country_id =
$request['country_id'];
            $user_profile->phone =
$request['phone'];
            $user_profile->avatar = $imagePath;
            $user_profile->save();
```

```
        } catch (\Exception $exception) {
            return response([
                'success' => false,
                'message' => 'Something went
wrong.',
                'error_message' => $exception-
>getMessage(),
            ]);
        }

        return response([
            'success' => true,
            'message' => 'User profile updated
successfully.',
            'data' => $user_profile
        ]);

    }

    public function viewProfile()
    {
        $user_profile = auth('api')->user()-
>userProfile;

        return response([
            'success' => true,
            'message' => 'User profile retrieved
successfully',
            'data' => $user_profile,
        ]);
    }
}
```

5.1.11. WebReview Controller

Lapisan ini bertugas untuk menghubungkan Web Review dengan database agar data yang telah diisikan pada form dapat tersimpan di database.

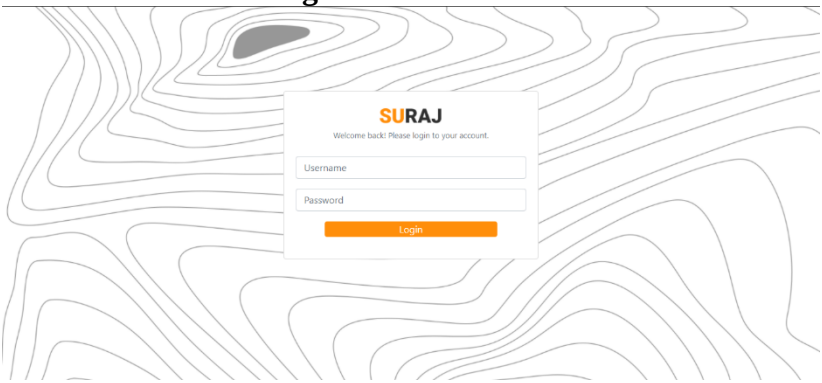
```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use App\WebReview;

class WebReviewController extends Controller
{
    public function index()
    {
        $web_reviews = WebReview::all();

        return response([
            'success' => true,
            'message' => 'Website reviews
retrieved successfully.',
            'data' => $web_reviews,
        ]);
    }

    public function store(Request $request)
    {
        try{
            $request->validate([
                'comment' => 'required|string'
            ]);
```

```
            $user_id = auth('api')->user()-
>getKey();

            $web_review = WebReview::create([
                'user_id' => $user_id,
                'comment' => $request['comment']
            ]);
        } catch (\Exception $exception){
            return response([
                'success' => false,
                'message' => 'Something went
wrong',
                'error_code' => $exception-
>getCode(),
                'error_message' => $exception-
>getMessage(),
            ]);
        }

        return response([
            'success' => true,
            'message' => 'Website review added
successfully.',
            'data' => $web_review,
        ]);
    }
}
```

5.2. Implementasi Antarmuka Pengguna

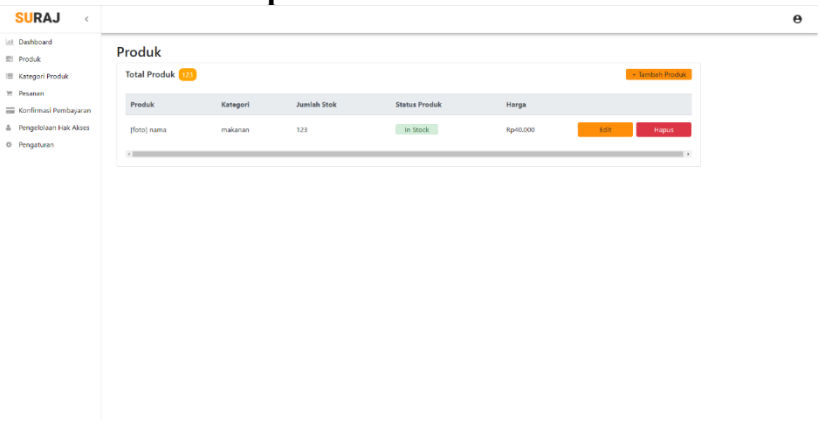
Pada bagian ini kami akan menampilkan antarmuka halaman yang ada pada Aplikasi Web E-commerce SURAJ.

5.2.1. Halaman login admin



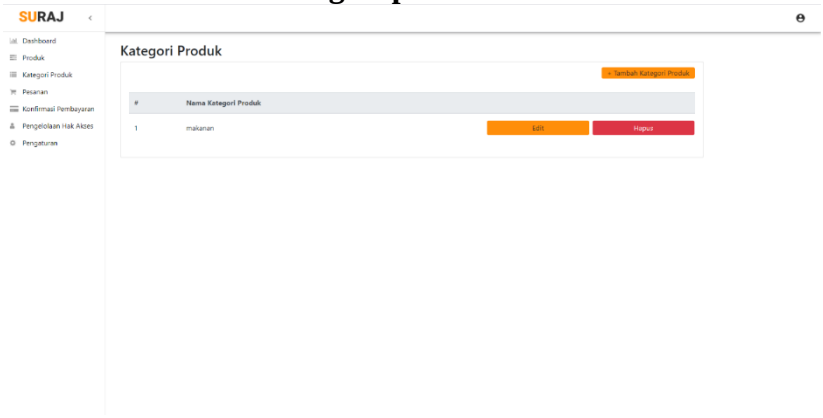
Gambar 2: Tampilan halaman login admin

5.2.2. Halaman produk



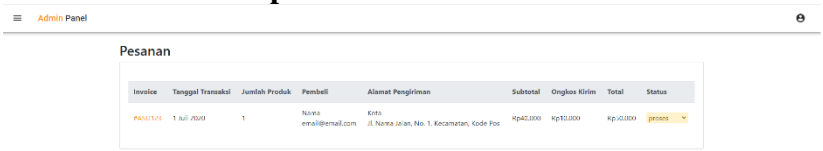
Gambar 3: Tampilan halaman produk

5.2.3. Halaman kategori produk



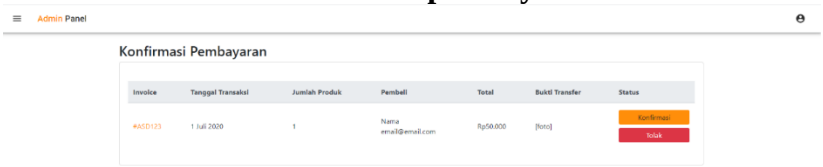
Gambar 4: Tampilan halaman kategori produk

5.2.4. Halaman pesanan



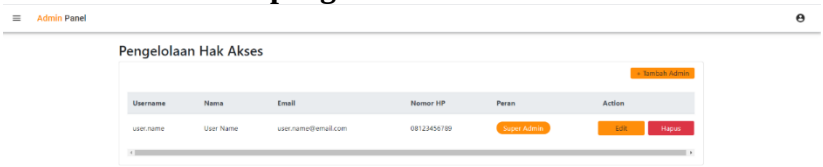
Gambar 5: Tampilan halaman pesanan

5.2.5. Halaman konfirmasi pembayaran



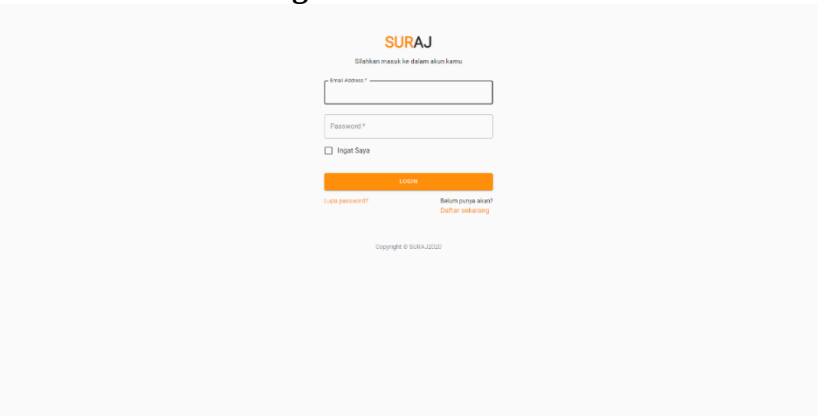
Gambar 6: Tampilan halaman konfirmasi pembayaran

5.2.6. Halaman pengelolaan hak akses



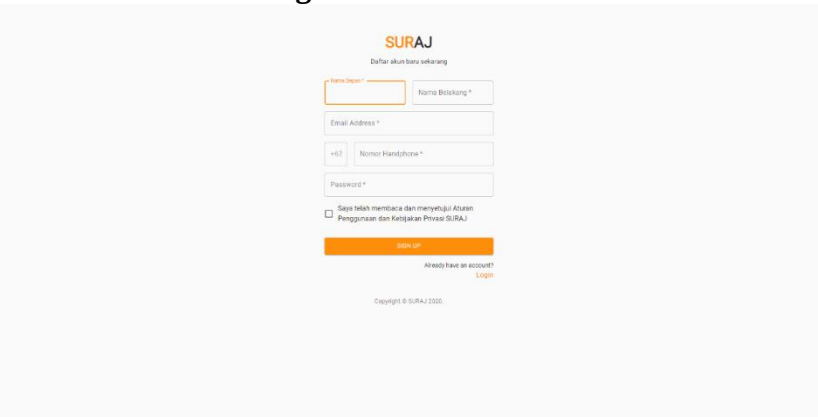
Gambar 7: Tampilan halaman pengelolaan hak akses

5.2.7. Halaman login user



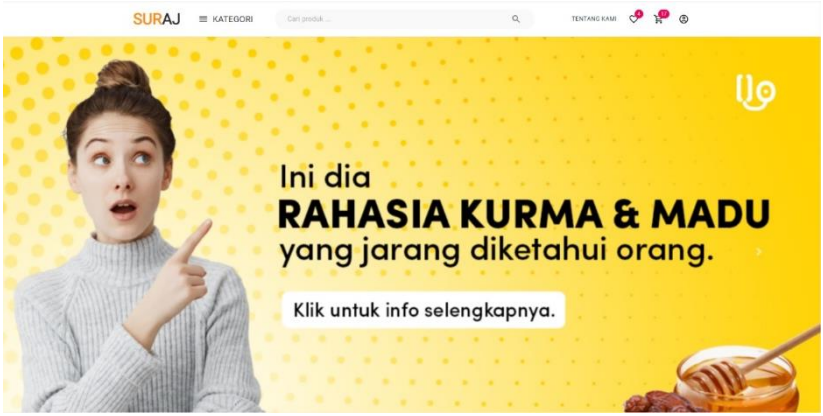
Gambar 8: Tampilan halaman login user

5.2.8. Halaman registrasi user

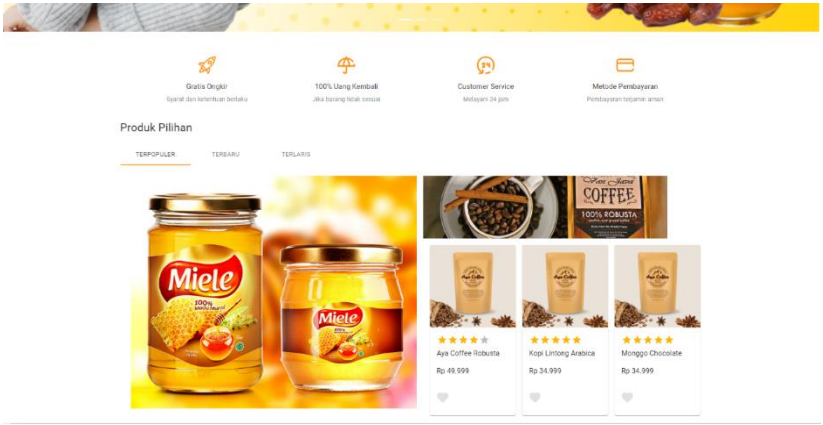


Gambar 9: Tampilan halaman registrasi user

5.2.9. Halaman utama

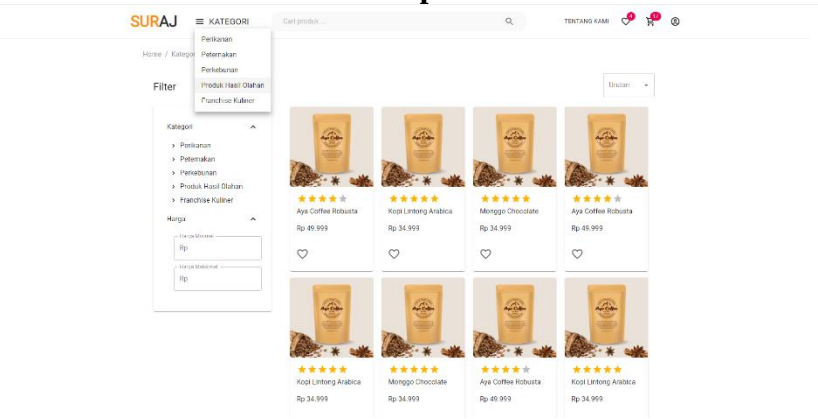


Gambar 10: Tampilan halaman utama bagian 1



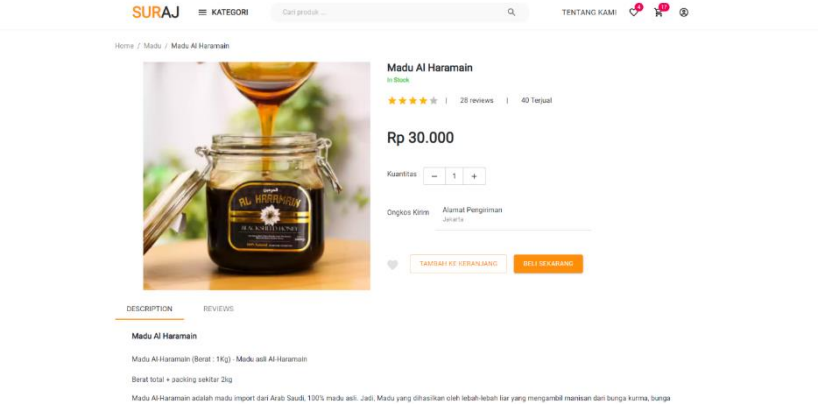
Gambar 11: Tampilan halaman utama bagian 2

5.2.10. Halaman daftar produk

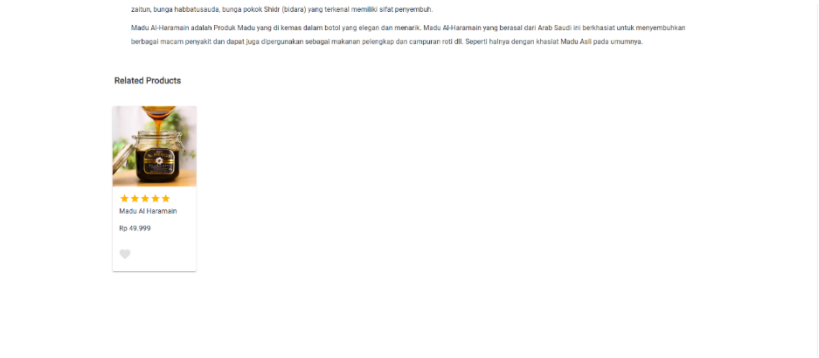


Gambar 12: Tampilan halaman daftar produk

5.2.11. Halaman detail produk

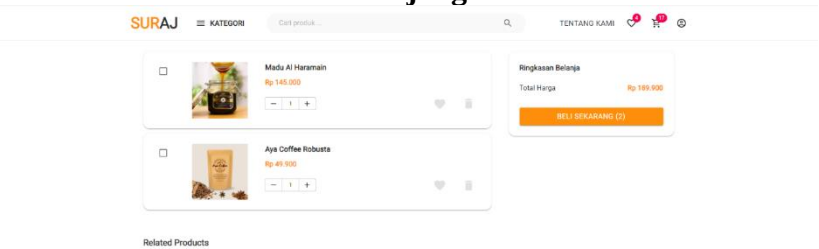


Gambar 13: Tampilan halaman detail produk bagian 1



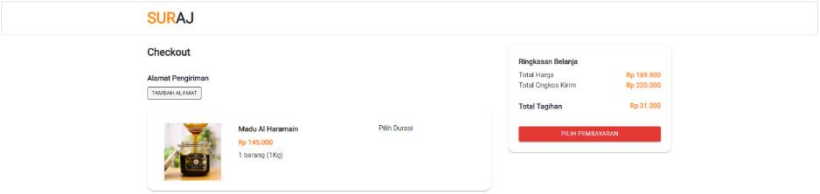
Gambar 14: Tampilan halaman detail produk bagian 2

5.2.12. Halaman keranjang

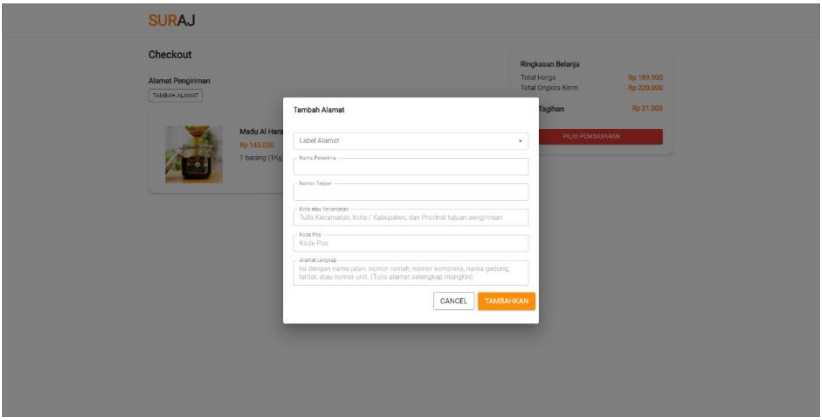


Gambar 15: Tampilan halaman keranjang

5.2.13. Halaman checkout



Gambar 16: Tampilan halaman checkout bagian 1



Gambar 17: Tampilan halaman checkout bagian 2

[Halaman ini sengaja dikosongkan]

BAB VI
PENGUJIAN DAN EVALUASI

Bab ini menjelaskan tahap uji coba terhadap Aplikasi Web E-commerce SURAJ. Pengujian dilakukan untuk memastikan kualitas perangkat lunak yang dibangun dan kesesuaian hasil eksekusi perangkat lunak dengan analisis dan perancangan perangkat lunak.

6.1. Skenario Pengujian

6.1.1. Melakukan login admin

1. Membuka halaman login untuk admin.
2. Memasukkan username dan password.
3. Menekan tombol login.

6.1.2. Melihat dashboard admin

1. Melakukan login admin.
2. Membuka halaman dashboard admin.

6.1.3. Menambah produk

1. Melakukan login admin.
2. Membuka halaman untuk menambahkan produk.
3. Memasukkan data produk yang dikehendaki.
4. Menekan tombol tambah.

6.1.4. Melihat daftar produk

1. Melakukan login admin.
2. Membuka halaman untuk melihat daftar produk.

6.1.5. Mengubah produk

1. Melakukan login admin.
2. Membuka halaman untuk melihat daftar produk.
3. Menekan tombol ubah pada produk yang dikehendaki.
4. Memasukkan data produk yang dikehendaki.
5. Menekan tombol simpan.

6.1.6. Menghapus produk

1. Melakukan login admin.
2. Membuka halaman untuk melihat daftar produk.
3. Menekan tombol hapus pada produk yang dikehendaki.

6.1.7. Melihat pemesanan

1. Melakukan login admin.
2. Membuka halaman untuk melihat daftar pemesanan.

6.1.8. Melihat transaksi

1. Melakukan login admin.
2. Membuka halaman untuk melihat daftar transaksi.

6.1.9. Mengonfirmasi pembayaran

1. Melakukan login admin.
2. Membuka halaman untuk melihat daftar transaksi.
3. Menekan tombol konfirmasi pada transaksi yang dikehendaki.

6.1.10. Memperbarui status pesanan

1. Melakukan login admin.
2. Membuka halaman untuk melihat daftar pemesanan.
3. Memilih status pesanan pada pemesanan yang dikehendaki.

6.1.11. Melihat laporan bulanan

1. Melakukan login admin.
2. Membuka halaman laporan bulanan.

6.1.12. Menambah admin

1. Melakukan login admin.
2. Membuka halaman untuk menambahkan admin.
3. Memasukkan data admin yang dikehendaki.
4. Menekan tombol tambah.

6.1.13. Melihat daftar admin

1. Melakukan login admin.
2. Membuka halaman untuk melihat daftar admin.

6.1.14. Mengubah admin

1. Melakukan login admin.
2. Membuka halaman untuk melihat daftar admin.
3. Menekan tombol ubah pada admin yang dikehendaki.
4. Memasukkan data admin yang dikehendaki.
5. Menekan tombol simpan.

6.1.15. Menghapus admin

1. Melakukan login admin.
2. Membuka halaman untuk melihat daftar admin.
3. Menekan tombol hapus pada admin yang dikehendaki.

6.1.16. Melakukan registrasi user

1. Membuka halaman registrasi untuk user.
2. Memasukkan data akun user yang dikehendaki.
3. Menekan tombol sign up.

6.1.17. Melakukan login user

1. Membuka halaman login untuk user.
2. Memasukkan email dan password.
3. Menekan tombol login.

6.1.18. Melakukan pencarian produk

1. Membuka halaman utama.
2. Memasukkan kata kunci pada search bar.
3. Menekan tombol search.

6.1.19. Melihat produk berdasarkan kategori

1. Membuka halaman utama.
2. Memilih kategori produk yang dikehendaki.

6.1.20. Melihat detail produk

1. Membuka halaman utama.
2. Memilih produk yang dikehendaki.

6.1.21. Melakukan chat

1. Membuka bubble chat.
2. Mengirimkan pesan.

6.1.22. Menambah produk ke keranjang

1. Melakukan login user.
2. Membuka halaman untuk melihat detail produk.
3. Mengisi jumlah produk yang dikehendaki.
4. Menekan tombol tambah ke keranjang.

6.1.23. Melihat produk di keranjang

1. Melakukan login user.
2. Membuka halaman keranjang.

6.1.24. Mengubah jumlah produk di keranjang

1. Melakukan login user.
2. Membuka halaman keranjang.
3. Memasukkan jumlah baru pada produk yang dikehendaki.

6.1.25. Menghapus produk dari keranjang

1. Melakukan login user.
2. Membuka halaman keranjang.
3. Menekan tombol hapus pada produk yang dikehendaki.

6.1.26. Melakukan checkout pemesanan

1. Melakukan login user.
2. Membuka halaman keranjang.
3. Memilih produk yang dikehendaki.

4. Menekan tombol lanjut.
5. Memasukkan data pengiriman yang dikehendaki.
6. Memasukkan data pembayaran yang dikehendaki.
7. Menekan tombol konfirmasi checkout.

6.1.27. Melacak proses pemesanan

1. Melakukan login user.
2. Membuka halaman untuk melacak proses pemesanan.

6.1.28. Menambah review produk

1. Melakukan login user.
2. Membuka halaman dari produk yang status pengirimannya sudah diterima.
3. Memasukkan rating dan komentar yang dikehendaki.

6.1.29. Mengubah review produk

1. Melakukan login user.
2. Membuka halaman dari produk yang status pengirimannya sudah diterima.
3. Mengubah rating dan/atau komentar yang dikehendaki.

6.1.30. Mengisi survey kepuasan layanan web

1. Melakukan login user.
2. Mengonfirmasi bahwa pengiriman produk sudah diterima.
3. Memasukkan komentar yang dikehendaki.

6.2. Evaluasi Pengujian

Pada subbab ini akan diberikan hasil evaluasi dari pengujian-pengujian yang telah dilakukan. Hasil evaluasi pengujian dapat dilihat pada Tabel 32.

Tabel 32: Hasil evaluasi pengujian aplikasi sesuai kebutuhan

No.	Kebutuhan	Status
UC001	Melakukan login admin	Berhasil
UC002	Melihat dashboard admin	Berhasil
UC003	Menambah produk	Berhasil
UC004	Melihat daftar produk	Berhasil
UC005	Mengubah produk	Berhasil
UC006	Menghapus produk	Berhasil
UC007	Melihat pemesanan	Berhasil
UC008	Melihat transaksi	Berhasil
UC009	Mengonfirmasi pembayaran	Berhasil
UC010	Memperbarui status pesanan	Berhasil
UC011	Melihat laporan bulanan	
UC012	Menambah admin	Berhasil
UC013	Melihat daftar admin	Berhasil
UC014	Mengubah admin	Berhasil
UC015	Menghapus admin	Berhasil
UC016	Melakukan registrasi user	Berhasil
UC017	Melakukan login user	Berhasil
UC018	Melakukan pencarian produk	Berhasil
UC019	Melihat produk berdasarkan kategori	Berhasil
UC020	Melihat detail produk	Berhasil

UC021	Melakukan chat	
UC022	Menambah produk ke keranjang	Berhasil
UC023	Melihat produk di keranjang	Berhasil
UC024	Mengubah jumlah produk di keranjang	Berhasil
UC025	Menghapus produk dari keranjang	Berhasil
UC026	Melakukan checkout pemesanan	Berhasil
UC027	Melacak proses pemesanan	
UC028	Menambah review produk	
UC029	Mengubah review produk	
UC030	Mengisi survey kepuasan layanan web	

[Halaman ini sengaja dikosongkan]

BAB VII
KESIMPULAN DAN SARAN

Kesimpulan yang didapat setelah melakukan pembuatan Aplikasi Web E-commerce SURAJ adalah sebagai berikut:

a. Aplikasi yang dibangun cukup sesuai dengan permintaan dan dapat dengan mudah dioperasikan oleh pengguna.
b. Dengan adanya Aplikasi Web E-commerce SURAJ, pengguna dapat melihat katalog produk yang ada di Sultan Ar-Rajabi serta melakukan pembelian pada produk-produk yang dikehendaki sedangkan pemilik perusahaan dapat mengelola produk serta pemesanan dan transaksi yang dilakukan oleh pengguna.

DAFTAR PUSTAKA

[1] Wikipedia (2020). Perdagangan elektronik – Wikipedia bahasa Indonesia, ensiklopedia bebas. [online] Available at: https://id.wikipedia.org/wiki/ Perdagangan_elektronik [Accessed 27 November 2020].
[2] Wikipedia (2020). HTML – Wikipedia bahasa Indonesia, ensiklopedia bebas [online] Available at: https://id.wikipedia.org/wiki/HTML [Accessed 27 November 2020].
[3] W3 (2020). Cascading Style Sheets. [online] Available at: https://www.w3.org/Style/CSS/Overview.en.html [Accessed 27 November 2020].
[4] Dewaweb (2017). Javascript – Dewaweb [online] Available at : https://www.dewaweb.com/blog/pengenalan-javascript/ [Accessed 27 November 2020].
[5] Niagahoster (2020). PHP – Niagahoster [online] Available at: https://www.niagahoster.co.id/blog/pengertian-php/ [Accessed 27 November 2020].
[6] Codepolitan (2017). Framework Laravel – Codepolitan [online] Available at: https://www.codepolitan.com/alasan-mengapa-kamu-harus-menggunakan-framework-laravel-5a08d435ddcfb [Accessed 27 November 2020].
[7] SOCS BINUS (2020). Apa itu React.Js? – SOCS BINUS [online] Available at: https://socs.binus.ac.id/2019/12/30/apa-itu-react-js/ [Accessed 27 November 2020].
[8] Herman Yuliansyah. 2014. Perancangan Replikasi Basis Data MySQL Dengan Mekanisme Pengamanan Menggunakan SSL Encryption. Jurnal Informatika. 8(1):826-836.

[9] Wikipedia (2020). Visual Studio Code – Wikipedia bahasa Inggris, ensiklopedia bebas [online] Available at: https://en.wikipedia.org/wiki/Visual_Studio_Code [Accessed 27 November 2020].

BIODATA PENULIS

Jeremy Vijay Wongso, lahir pada tanggal 30 Juni 1999 di Surabaya. Penulis merupakan mahasiswa yang sedang menempuh studi di Departemen Informatika Institut Teknologi Sepuluh Nopember (ITS).

Pristi Zahara, lahir pada tanggal 26 April 2000 di Tangerang. Penulis merupakan mahasiswa yang sedang menempuh studi di Departemen Informatika Institut Teknologi Sepuluh Nopember (ITS).

Rana Wijdan Naim, lahir pada tanggal 15 Juni 1998 di Nganjuk. Penulis merupakan mahasiswa yang sedang menempuh studi di Departemen Informatika Institut Teknologi Sepuluh Nopember (ITS).